Stress Testing Technologies for Citrix® MetaFrame®

Michael G. Norman, CEO
December 5, 2001

Contents

Executive Summary

Scapa® Technologies, a Citrix Business Alliance member, has developed *n-Tier Resonance* a unique technology which, for the first time, makes it possible to run a reliable and accurate stress test of applications deployed using the popular Citrix MetaFrame technology.

Introduction

Citrix Systems Inc, through its Citrix MetaFrame server and ICA architecture has provided a powerful enabling technology for server-based computing. Those who run applications on servers, with or without a formal service level agreement, really ought to ensure that their architecture is actually capable of handling the application load. This is achieved by a process known as *stress testing*.

Approaches to Stress Testing

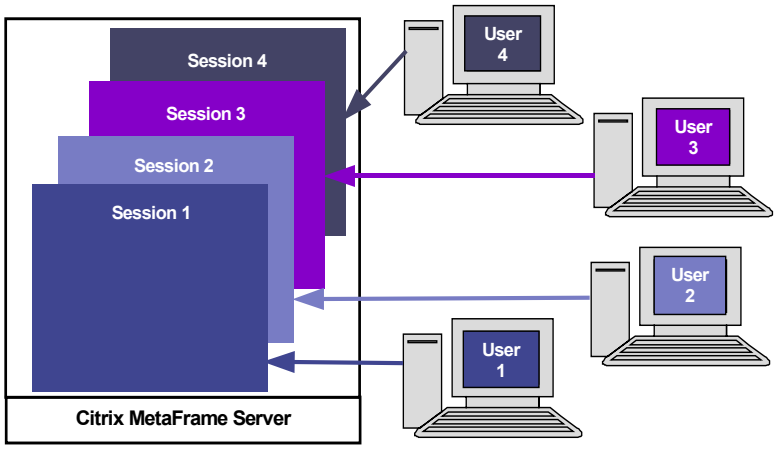
Stress testing is performed by a software tool that *pretends* to be users of a system. It is necessary to control how many users there are, what they are doing and how often they are doing it. The tool measures the time it takes for the application to respond to the user and identifies any stress-related application failures. For capacity planning purposes the tool measures how things scale as the number of servers is increased and can check the efficiency of load balancing. During tests, monitors are set on various system parameters (like memory, disk, network and processor usage), and problems can be fixed by making changes to system configurations and applications.

To get these tests to work it is necessary to introduce some variability amongst the simulated users. For example, multiple users of an application like Microsoft® Word cannot simultaneously write to the same document, or when stress testing a business application it is usually necessary for different users to be dealing with different products or customers to mimic real database locking patterns.

Windows Applications

So, let us think about stress testing Citrix MetaFrame applications. To pretend to be a user, the test software needs to click buttons and type in text at the GUI. There is a wide range of GUI scripting tools (such as Rational Software's Visual Test) which were developed for functional testing of Windows applications and which simulate user activity at a GUI. Multiple copies of these tools can be run simultaneously to simulate multiple users.

One problem arises with using these tools. Real users wait for buttons to appear before they click them. Simulated users need to do the same because if a button is clicked before it has been drawn either nothing happens or something completely unexpected happens, but the right result certainly does not happen. All subsequent button clicks are then out of sequence. In fact, *the test simply*



breaks. To help resolve these problems GUI scripting tools can wait for things like buttons or menus to appear so that they interact with the application at the right time. It is a bit complex but with practice effective robust tests can be built.

This is not Windows®

Citrix MetaFrame allows applications that were built for a desktop environment to run on the server (as shown in the diagram above) without significant modification, using a standard component known as the ICA client on the desktop computer. Citrix MetaFrame essentially inserts itself between an application and its user's screen, keyboard and mouse, diverting data to and from another computer somewhere across the network.

ICA makes a clean separation between the *presentation tier* of an architecture which runs on the client and the *application tier* which runs at the server. Significantly, ICA does not operate at the level of Windows buttons and menus, but at the lower-level presentation events that make up those objects. To use a surrealist analogy, Citrix MetaFrame does not send *Windows* to the client it sends *Pictures of Windows*.

Testing an ICA Architecture

There are two basic approaches to stress testing Citrix ICA architectures: pretending to be users at the *presentation tier* on the client or at the *application tier* on the server. Both have their disadvantages.

Application Tier

To test at the application tier, lots of copies of a GUI scripting tool can be run on the server to simulate users of the application, and whilst this cannot be done directly, it is possible to run lots of ICA client sessions each of which is running a GUI scripting tool. Citrix MetaFrame supplies a test kit based upon this approach. Multiple ICA sessions can be run from a given client machine so it is not necessary to have a huge amount of client hardware to stress test a server.

One possible objection to this approach is that the GUI scripting tool is itself a program. It takes up resources on the server alongside the application it is running. This introduces inaccuracies into the test results. Note however that the GUI scripting tool always *adds* to the load on the server creating an *underestimate* of the capacity of the server, so we are at least operating on the side of caution. It is possible to measure the effect using standard Windows management options and a good GUI scripting tool will typically consume less than 2% of the resources that the application consumes, so it can usually be ignored.

The problem with application-tier testing in the Citrix MetaFrame environment is that *the user experience is not taken into consideration*. The presentation tier controls the application tier and not the other way around. If you introduce stress at the application tier, the tests run onwards regardless of any event happening at the presentation tier and of any delays that the communication latency between client and server would introduce in a real environment.

Given the use of Citrix MetaFrame in wide area deployments and across the public Internet this limitation renders any timing data unfit for most purposes. It also limits the use of server capacity data, because latency skews the load profile on the server and changes a large number of system parameters like memory usage and the rate of context switching.

Presentation Tier

To resolve the inherent inaccuracies of server-end testing it would seem natural to test at the presentation tier by running a GUI scripting tool at the client-end. However this runs into a crucial problem. The simulated user has to wait for windows etc. to appear, but all that Citrix MetaFrame sends are *pictures of windows*. The scripts have to be written to wait for these pictures to appear by polling a screen buffer comparing it with a bit-map that is expected to arise. Polling for bit-maps is bad news for a wide range of reasons.

Choosing a bit-map. The trick is to wait for something that is guaranteed to appear just before it becomes possible to successfully click the button. The problem is to identify a piece of the screen guaranteed to be in a particular state at the point at which this becomes possible. If it goes wrong the test breaks.

Inconsistency. Imagine a script that clicks a button to cause a menu to pop up. The script then clicks within the menu, without actually waiting for the menu to arrive. This script is tested in a single-user environment and it works because the menu is always there before the second click is issued. A complex stress test is then set up running multiple copies of the script simultaneously. This loads up the server so the graphics stream slows down so that when the script makes the second click the menu has not yet appeared. The test breaks.

Variability is difficult. The problem is that variability can change the output to the screen. For example, the application may display the user's name in a list box. One of the user name variants may be long enough to cause a scroll bar to appear where none appeared in the original script which causing the screen

layout to be rearranged. The bit-map comparison may become invalid and the test breaks.

Too much test hardware is required. To detect a bit-map, each script needs to watch its user interface and periodically check it for a particular set of pixels. This takes up client-end memory and CPU and can lead to a requirement for a bigger machine for stress testing than the server itself. In addition, to check a bit-map reliably, the corresponding window has to be permanently in the foreground on the client machine, which needs a very large screen resolution.

The timings are wrong in any case. Bit-map polling does not indicate when the bit-map appeared. It indicates the first moment after the bit-map appeared that it was polled for. If there are a lot of scripts running on the same client machine, this is not a good approximation to the actual time that the bit-map arrived.

n-Tier Resonance

Scapa Technologies has developed a unique n-Tier Resonance technology that resolves the synchronisation problems of stress testing in tiered architectures like Citrix ICA.

Stress that is introduced at one tier in the architecture is made to resonate accurately through other tiers of the architecture so that it becomes possible for server loading patterns and client-end timings to reflect the effects of communications latency. The resonance is achieved by binding together tiers of the architecture using low-level synchronisation logic.

Scapa StressTest for Citrix MetaFrame

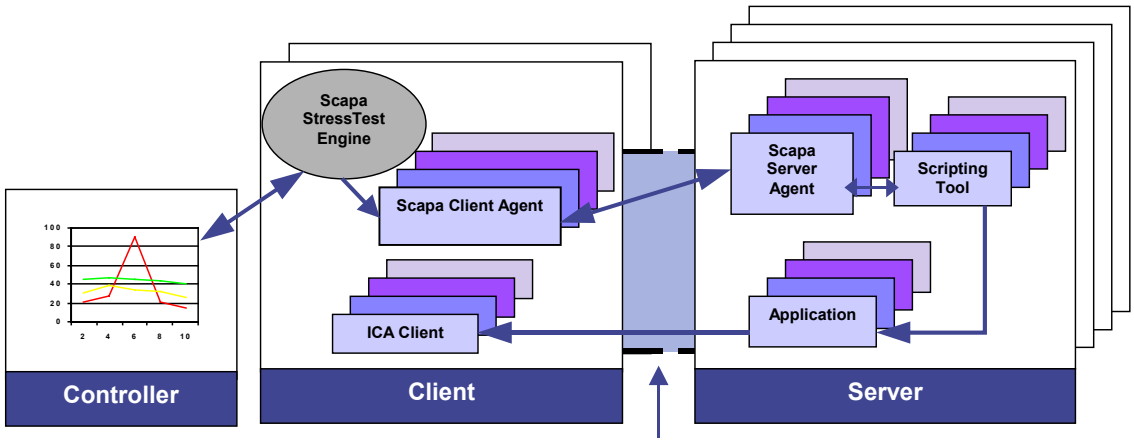
n-Tier Resonance is the basis of Scapa StressTest for Citrix MetaFrame, a full-function stress testing tool and the only effective solution for stress testing Citrix MetaFrame applications.

Tests are run at the server using a standard GUI scripting tool. The tests send synchronization tokens to the client alongside the graphics stream via the *ICA virtual channel.* Software at the client end identifies the tokens and does not need to deal with bit-maps. It issues tokens back to the server along the ICA Virtual Channel, which control the simulation of input events in the server-end script.

The outgoing token is so small in comparison to the graphics stream that it does not skew the test results. The return token takes up a similar amount of ICA bandwidth as the keyboard or mouse click that it replaces. The overall architecture is shown in the diagram below.

Architecture

The capture and editing capabilities of a standard GUI scripting tool are used to build one or more user test scripts. Synchronisation statements are inserted to connect them by way of the Scapa StressTest open interface to a Scapa Server Agent. The Scapa Script Activator is then used to automatically construct the resonance control logic which manages the Scapa Client Agents. These are



distributed amongst one or more client machines under the control of the StressTest engine and a GUI on a Controller machine.

Features

Scapa StressTest is an established stress testing technology. In addition to dedicated Citrix MetaFrame connectivity it has a number of key features:

Dynamic control of load: The load on the servers can be controlled through a single user interface via one or more sliders. The user interface is like a graphic equaliser, which can crank up the load and see where the system breaks, whilst watching third-party monitoring tools. It is also possible to reduce user "think time", dynamically changing the time between users' mouse and keyboard clicks.

Control over business throughput as well as user count. If it is important to know the number of transactions per second the system can support rather than the number of users, StressTest allows this to be controlled directly.

Predefined tests and scheduled tests execution. If *dynamic* control is not required, tests can be pre-defined to run in certain ways, for example for regression testing. They can also be scheduled for out of hours execution.

Distributed test execution. Tests can be run from multiple client machines, to measure end-user performance in the various geographies over which the applications are being deployed.

Test result analysis. All the data collected during a test is available for analysis with Scapa StressTest reporting features and third party reporting tools.

Next Steps

This White Paper is a response to the challenges presented to the testing industry by the success of Citrix Solutions in delivering the promise of server-based computing. Readers are encouraged to refer to further information about other stress testing technologies at www.scapatech.com or by sending an e-mail enquiry to *contact@scapatech.com.*

Scapa Technologies Inc.
245 Park Avenue
39th Floor
NYC, NY 10167

Tel: +1 212 792 4032
Fax: +1 212 372 8798

Scapa Technologies Limited
125 McDonald Road,
Edinburgh
EH7 4NW, Scotland

Tel: +44 131 652 3939
Fax: +44 131 652 3299

www.scapatech.com

contact@scapatech.com

Part Number: WP/011/1101_BetaMinus